ONELAB: Open Numerical Engineering LABoratory

C. Geuzaine

University of Liège, Belgium

Joint work with R. Sabariego, P. Dular, J.-F. Remacle, E. Marchandise, F. Henrotte

EMF2013 - April 23 2013

Some Background...

- We write quite a lot of codes, mostly solvers (GetDP & co)... as most of you do
- Started Gmsh mesh generator & pre-/post-processor in 1996 as a hobby, to “scratch an itch”
- First public binary release in 1998
- GNU GPL release in 2003
- “Getting serious” in 2007: publications, new algorithms, growing community

Some Background...

Today, Gmsh represents

- > 250k lines of code
- Still only 2 core devs; 75 devs with repo write access
- > 700 people on mailing list
- > 1000 binary downloads per week
- > 200 citations in 2012

Some Background...

Today, Gmsh represents

- > 250k lines of code
- Still only 2 core devs; 75 devs with repo write access
- > 700 people on mailing list
- > 1000 binary downloads per week
- > 200 citations in 2012

Were are we headed

- Stable C++ and Python API (currently we know of about 15 solvers relying on the undocumented API)
- ONELAB

General Goal of the ONELAB Project

Develop a platform for integrating free Finite Element Analysis (FEA) software

General Goal of the ONELAB Project

Develop a platform for integrating free Finite Element Analysis (FEA) software

Economic rationale:

- Prohibitive cost of commercial FEA codes for many use-cases (SMEs, education)

General Goal of the ONELAB Project

Develop a platform for integrating free Finite Element Analysis (FEA) software

Economic rationale:

- Prohibitive cost of commercial FEA codes for many use-cases (SMEs, education)

Two difficulties associated with free FEA software :

- **Heterogeneity of the tools (1)**
- **Missing "expert" layer and documentation (2)**

Existing Projects

The closest (probably): SALOME (http://salome-platform.org)

Main differences design-wise:

- Should be “fast” and “light” -- Emphasis on simplicity
- Provide *abstract* interfaces to CAD, solvers and post-processing
- Optional GUI

Quite close: GiD (http://gid.cimne.upc.es), but not free

ONELAB Guiding Principles

- Don’t reimplement, interface
- Make it easy to provide templates, with interactive parameter modification
- Make it as small and as easy to maintain as possible (no solver-dependent code in the interface)

ONELAB Guiding Principles

- Don’t reimplement, interface
- Make it easy to provide templates, with interactive parameter modification
- Make it as small and as easy to maintain as possible (no solver-dependent code in the interface)

ONELAB role = data centralization, (optional) modification and redispatching

Issues of completeness and consistency of the parameter set are completely dealt with on the solver side

ONELAB Features

ONELAB Features

(1) Heterogeneity of the tools

(2) Missing “expert” layer and documentation

ONELAB Features

(1) Abstract interface to FEA codes

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing
- Implemented in Gmsh:

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing
- Implemented in Gmsh:
    - Parameter exchange library

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing
- Implemented in Gmsh:
  - Parameter exchange library
  - C++ and Python clients

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing
- Implemented in Gmsh:
    - Parameter exchange library
    - C++ and Python clients

(2) Development and documentation of templates ("meta-models")

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing
- Implemented in Gmsh:
    - Parameter exchange library
    - C++ and Python clients

(2) Development and documentation of templates ("meta-models")

- Model: backbox, parameterizable via abstract interface

ONELAB Features

(1) Abstract interface to FEA codes

- CAD & meshing; physical properties, constraints & code drivers; post-processing
- Implemented in Gmsh:
    - Parameter exchange library
    - C++ and Python clients

(2) Development and documentation of templates ("meta-models")

- Model: backbox, parameterizable via abstract interface
- Meta-model: set of models + selection logic

Demo

Demo

Tadaa!

ONELAB Implementation

Client-server:

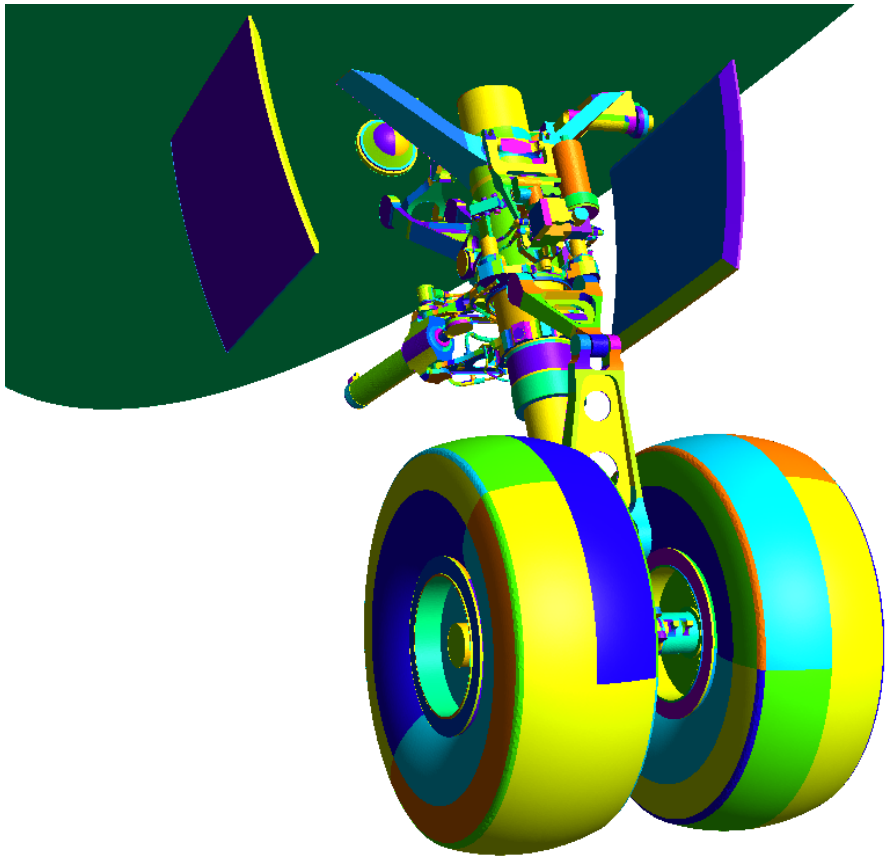
- Clients: CAD kernels, meshers, solvers, post-processors
- Server: Gmsh + database

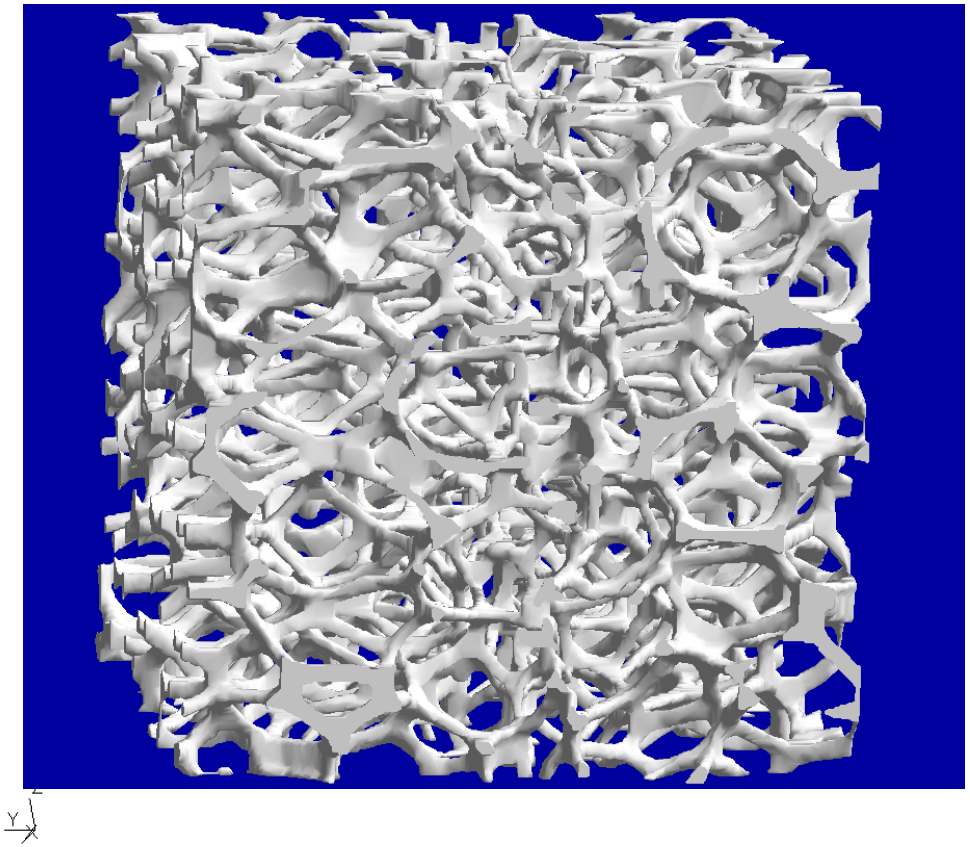
Abstract interface:

- The server **has no a priori knowledge of the clients** (no meta-language or exchange file format)
- The server **does not write input files for (native) clients**: the client communicates with the server to define what information should be exchanged

ONELAB Implementation

Abstract CAD & meshing interface = Gmsh's `GModel` B-Rep model

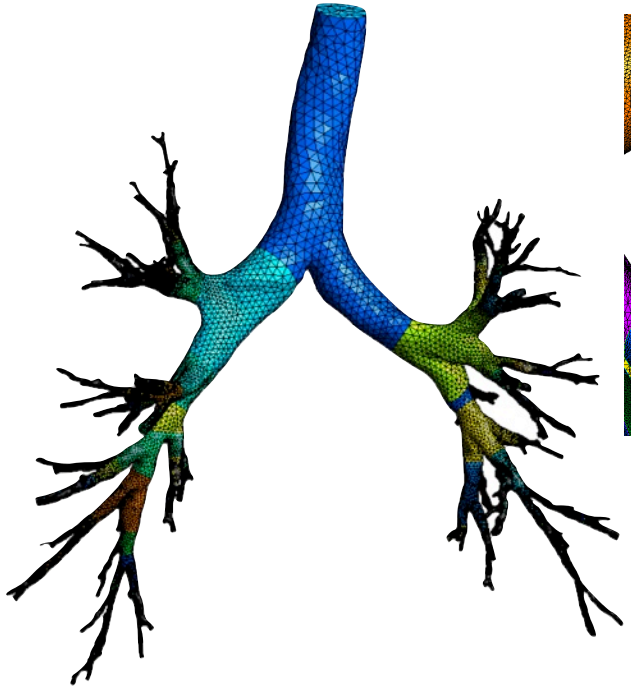


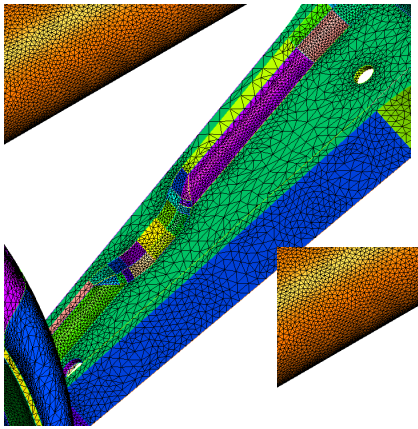


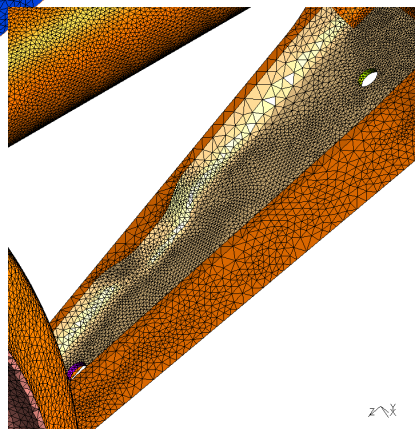
ONELAB Implementation

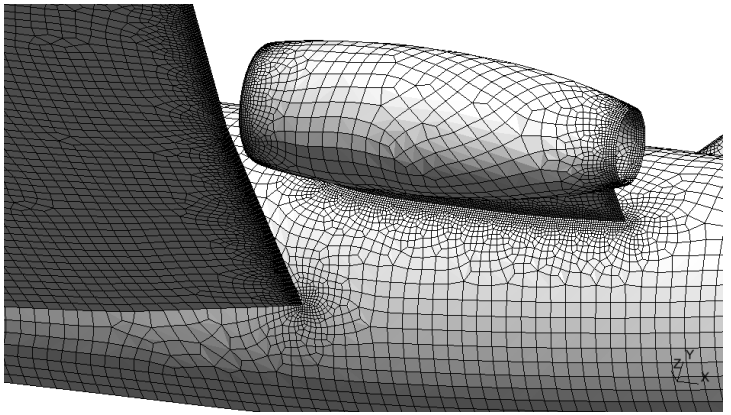
Abstract CAD & meshing interface = Gmsh's `GModel` B-Rep model

- Direct access to underlying CAD kernels (no translations)
- New features: hybrid models, remeshing, cross-patch meshing, quad meshes, curved meshes









ONELAB Implementation

Abstract interface to physical properties, constraints & code drivers:

- Library for parameter exchange:
  - Server in C++ for portability, e.g. on iOS/Android (`onelab::server`)
  - Client in C++ (`onelab::client`) or Python
  - Exchange parameters (`onelab::parameter`) through TCP/IP or in-memory

ONELAB Implementation

Abstract interface to physical properties, constraints & code drivers:

- Library for parameter exchange:
  - Server in C++ for portability, e.g. on iOS/Android (`onelab::server`)
  - Client in C++ (`onelab::client`) or Python
  - Exchange parameters (`onelab::parameter`) through TCP/IP or in-memory
- "Native" clients use C++ (e.g. GetDP in the demo) or Python directly
- "Interfaced" clients use Python, by instrumenting their input files (API available in a couple of month on http://onelab.info)
  - Currently: Elmer, OpenFOAM, Code_Aster, Abaqus, Gnuplot

ONELAB Implementation

`onelab::parameter`

- name as ‘/’-separated path
- dynamic dependency list of clients and status change
- decorations (help, bounds, choices, ...)
- serialization and deserialization

ONELAB Implementation

`onelab::parameter`

- name as ‘/’-separated path
- dynamic dependency list of clients and status change
- decorations (help, bounds, choices, ...)
- serialization and deserialization

Example for native Gmsh & GetDP clients (in .geo or .pro files):

```
DefineConstant[ N = {32, Label “Number of slices”} ];
```

Example for Python client:

```
c = onelab.client()
N = c.defineNumber('Number of slices', 32)
```

ONELAB Implementation

Abstract interface to post-processing:

- "Native": based on Gmsh's `PView`, `PViewData`, `PViewOptions`
  - Generic operations on list-based or GModel-based datasets
  - Adaptive visualization of high-order datasets by explicit specification of interpolation matrices

$$P\ (d \times 3) \qquad p_i = \xi^{P_{i1}} \eta^{P_{i2}} \zeta^{P_{i3}}$$

$$\Phi\ (d \times d) \qquad \phi_i = \sum_{k=1}^{d} p_k \Phi_{ik}$$

- "Interfaced": same as interface to other clients

Conclusion

ONELAB is:

- A simple (trivial?) way to interface FEA solvers
- Interactive, based on Gmsh, and free

- You can try it out with native (C++) clients now:
  - Gmsh for CAD/meshing
  - GetDP for electromagnetics and some “multiphysics”
- Generic interface for other solvers using Python will be available soon (later this year)

http://onelab.info